GLAD package: Gain and Loss Analysis of DNA

Philippe Hupe[1,2] and Emmanuel Barillot[2]

May 8, 2009

1. UMR 144 CNRS/Institut Curie, Institut Curie, 26, rue d'Ulm, Paris, 75248 cedex 05, France
2. Service Bioinformatique, Institut Curie, 26, rue d'Ulm, Paris, 75248 cedex 05, France
glad@curie.fr http://bioinfo.curie.fr

Contents

1 Overview

This document presents an overview of the `GLAD` package (Gain and Loss Analysis of DNA). This package is devoted to the analysis of Array Comparative Genomic Hybridization (array CGH) (Pinkel et al., 1998; Snijders et al., 2001; Solinas-Toldo et al., 1997; Ishkanian et al., 2004). The methodology for detecting the breakpoints delimiting altered regions in genomic patterns and assigning a status (normal, gained or lost) to each chromosomal region described in the paper Hupé et al. (2004) is implemented in this package. Some graphical functions are provided as well.

2 Data

2.1 Public data set

We used the public data set described in Snijders et al. (2001). The data corresponds to 15 human cell strains with known karyotypes (12 fibroblast cell strains, 2 chorionic villus cell strains, 1 lymploblast cell strain) from the NIGMS Human Genetics Cell Repository (`http://locus.umdnj.edu/nigms`). Each cell strain has been hybridized on CGH arrays of 2276 BACs, spotted in triplicates. Two array CGH profiles from the data obtained by Veltman et al. (2003) are available.

2.2 Bladder cancer data

Bladder cancer data from tumors collected at Henri Mondor Hospital (CrÃl'teil, France) (Billerey et al., 2001) have been hybridized on CGH arrays composed of 2464 BACs (Radvanyi, Pinkel et al., unpublished results). In this data, only the log-ratios are provided and no information about clones is available since the data is not yet published. This data allows only some graphical functionalities to be shown and will be used as a support to illustrate some functions for array normalization (not yes available in the current version of the package).

3 GLAD classes

3.1 arrayCGH

This class stores raw values after images analysis. The object arrayCGH is a list with at least a data.frame named arrayValues and a vector named arrayDesign. The data.frame arrayValues must contain the following fields:

Col Vector of columns coordinates.

Row Vector of rows coordinates.

... Other elements can be added.

The vector arrayDesign is composed of 4 values: c(arrayCol, arrayRow, SpotCol, SpotRow). The array CGH is represented by arrayRow*arrayCol blocs and each bloc is composed of SpotRow*SpotCol spots. N.B.: Col takes the values in 1:arrayRow*SpotRow and Row takes the values in 1:arrayCol*SpotCol

3.2 profileCGH and profileChr

This class stores synthetic values related to each clone available on the arrayCGH. The object profileChr corresponds to data of only one chromosome. Objects profileCGH and profileChr are composed of a list with the first element profileValues which is a data.frame with the following columns names:

LogRatio Test over Reference log-ratio.

PosOrder The rank position of each clone on the genome.

PosBase The base position of each clone on the genome.

Chromosome Chromosome name.

Clone The name of the corresponding clone.

... Other elements can be added.

LogRatio, Chromosome and PosOrder are compulsory.
To create those objects you can use the function *as.profileCGH*.

4 Analysis of array CGH profile

Two functions are available: *glad* and *daglad*. The second one is an improvment of of first one which was originally describes in Hupé et al. (2004). We recommand to use the *daglad* function. For fast computation use the option *smoothfunc=haarseg*.

4.1 Segmentation algorithms

Two algorithms are available for data segmentation:

- AWS (Polzehl and Spokoiny, 2000, 2002)
- HaarSeg (Ben-Yaacov and Eldar, 2008)

4.2 The *glad* function

A result of the GLAD methodology on cell line gm13330 (Snijders et al., 2001) is presented in **Figure 1**.

```
[1] ""
[1] "Have fun with GLAD"
[1] "For smoothing it is possible to use either"
[1] "the AWS algorithm (Polzehl and Spokoiny, 2002)"
[1] "or the HaarSeg algorithm (Ben-Yaacov and Eldar, Bioinformatics,  2008)"
[1] ""
[1] "If you use the package with AWS, please cite:"
[1] "Hupe et al. (Bioinformatics, 2004) and Polzehl and Spokoiny (2002)"
[1] ""
[1] "If you use the package with HaarSeg, please cite:"
[1] "Hupe et al. (Bioinformatics, 2004) and (Ben-Yaacov and Eldar, Bioinformatics, 2008)"
[1] ""
[1] "For fast computation it is recommanded to use"
[1] "the daglad function with smoothfunc=haarseg"
[1] ""
```

```
> data(snijders)
> profileCGH <- as.profileCGH(gm13330)
> res <- glad(profileCGH, mediancenter = FALSE, smoothfunc = "lawsglad",
```

```
+     bandwidth = 10, round = 1.5, model = "Gaussian", lkern = "Exponential",
+     qlambda = 0.999, base = FALSE, lambdabreak = 8, lambdacluster = 8,
+     lambdaclusterGen = 40, type = "tricubic", param = c(d = 6),
+     alpha = 0.001, msize = 5, method = "centroid", nmax = 8,
+     verbose = FALSE)

[1] "Smoothing for each Chromosome"
[1] "Optimization of the Breakpoints"
[1] "subsetdata"
   user  system elapsed
      0       0       0
[1] "diff"
Time difference of 0.001884937 secs
[1] "Temps findCluster: 0.0898079872131348"
[1] "subsetdata"
   user  system elapsed
      0       0       0
[1] "diff"
Time difference of 0.00165987 secs
[1] "Temps findCluster: 0.0220899581909180"
[1] "subsetdata"
   user  system elapsed
  0.000   0.000   0.001
[1] "diff"
Time difference of 0.001779795 secs
[1] "Temps findCluster: 0.0231268405914307"
[1] "subsetdata"
   user  system elapsed
      0       0       0
[1] "diff"
Time difference of 0.001745939 secs
[1] "Temps findCluster: 0.092634916305542"
[1] "subsetdata"
   user  system elapsed
  0.000   0.000   0.001
[1] "diff"
Time difference of 0.001667023 secs
[1] "Temps findCluster: 0.0221509933471680"
[1] "subsetdata"
   user  system elapsed
      0       0       0
[1] "diff"
Time difference of 0.001660109 secs
[1] "Temps findCluster: 0.0229539871215820"
[1] "subsetdata"
   user  system elapsed
  0.000   0.000   0.001
[1] "diff"
Time difference of 0.001662016 secs
[1] "Temps findCluster: 0.0221960544586182"
```

```
[1] "subsetdata"
   user  system elapsed
      0       0       0
[1] "diff"
Time difference of 0.001662970 secs
[1] "Temps findCluster: 0.0228350162506104"
[1] "subsetdata"
   user  system elapsed
      0       0       0
[1] "diff"
Time difference of 0.001672029 secs
[1] "Temps findCluster: 0.0222229957580566"
[1] "subsetdata"
   user  system elapsed
  0.004   0.000   0.000
[1] "diff"
Time difference of 0.001662970 secs
[1] "Temps findCluster: 0.0227220058441162"
[1] "subsetdata"
   user  system elapsed
  0.000   0.000   0.001
[1] "diff"
Time difference of 0.001682997 secs
[1] "Temps findCluster: 0.0222489833831787"
[1] "subsetdata"
   user  system elapsed
      0       0       0
[1] "diff"
Time difference of 0.001660109 secs
[1] "Temps findCluster: 0.0230309963226318"
[1] "subsetdata"
   user  system elapsed
      0       0       0
[1] "diff"
Time difference of 0.001667023 secs
[1] "Temps findCluster: 0.0221419334411621"
[1] "subsetdata"
   user  system elapsed
      0       0       0
[1] "diff"
Time difference of 0.001657009 secs
[1] "Temps findCluster: 0.0226471424102783"
[1] "subsetdata"
   user  system elapsed
  0.000   0.000   0.001
[1] "diff"
Time difference of 0.001776934 secs
[1] "Temps findCluster: 0.0224220752716064"
[1] "subsetdata"
   user  system elapsed
```

```
      0       0       0
[1] "diff"
Time difference of 0.001660109 secs
[1] "Temps findCluster: 0.0228810310363770"
[1] "subsetdata"
   user  system elapsed
      0       0       0
[1] "diff"
Time difference of 0.001673937 secs
[1] "Temps findCluster: 0.0222041606903076"
[1] "subsetdata"
   user  system elapsed
      0       0       0
[1] "diff"
Time difference of 0.001664877 secs
[1] "Temps findCluster: 0.0229701995849609"
[1] "subsetdata"
   user  system elapsed
      0       0       0
[1] "diff"
Time difference of 0.001663923 secs
[1] "Temps findCluster: 0.0220668315887451"
[1] "subsetdata"
   user  system elapsed
      0       0       0
[1] "diff"
Time difference of 0.001662016 secs
[1] "Temps findCluster: 0.022629976272583"
[1] "subsetdata"
   user  system elapsed
  0.000   0.000   0.001
[1] "diff"
Time difference of 0.001666069 secs
[1] "Temps findCluster: 0.0220551490783691"
[1] "subsetdata"
   user  system elapsed
  0.004   0.000   0.000
[1] "diff"
Time difference of 0.001655102 secs
[1] "Temps findCluster: 0.0226831436157227"
[1] "subsetdata"
   user  system elapsed
      0       0       0
[1] "diff"
Time difference of 0.00166893 secs
[1] "Temps findCluster: 0.0220520496368408"
[1] "subsetdata"
   user  system elapsed
  0.000   0.000   0.001
[1] "diff"
```

```
Time difference of 0.003546953 secs
[1] "Temps findCluster: 0.168665170669556"
[1] "Results Preparation"
```

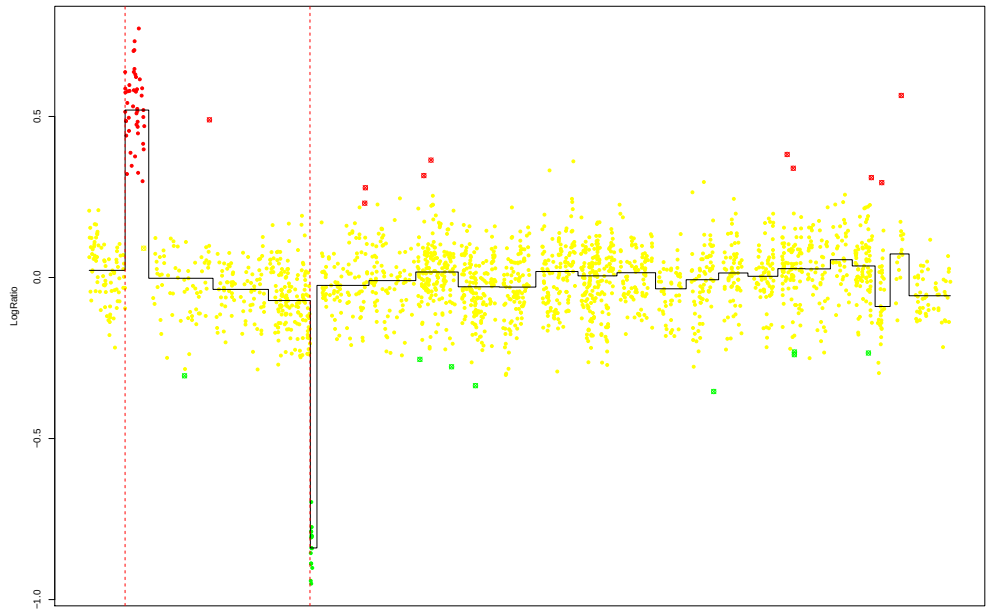

Figure 1: Results of glad on cell line gm13330 (Snijders data).

4.3 The *daglad* function

The algorithm implemented in this function is a slightly modified version of the GLAD algorithm.

```
> data(veltman)
> profileCGH <- as.profileCGH(P9)
> profileCGH <- daglad(profileCGH, mediancenter = FALSE, normalrefcenter = FALSE,
+     genomestep = FALSE, smoothfunc = "lawsglad", lkern = "Exponential",
+     model = "Gaussian", qlambda = 0.999, bandwidth = 10, base = FALSE,
+     round = 1.5, lambdabreak = 8, lambdaclusterGen = 40, param = c(d = 6),
+     alpha = 0.001, msize = 5, method = "centroid", nmin = 1,
+     nmax = 8, amplicon = 1, deletion = -5, deltaN = 0.2, forceGL = c(-0.3,
+         0.3), nbsigma = 3, MinBkpWeight = 0.35, CheckBkpPos = TRUE)

[1] "Smoothing for each Chromosome"
[1] "Optimization of the Breakpoints"
[1] "DNA copy number calling"
[1] "subsetdata"
   user  system elapsed
      0       0       0
[1] "diff"
Time difference of 0.002765179 secs
[1] "Temps findCluster: 0.174000024795532"
[1] "jointure BkpInfo"
   user  system elapsed
  0.004   0.000   0.002
[1] "Check Breakpoints Position"
[1] "Results Preparation"
```

The *daglad* function allows to choose some threshold to help the algorithm to identify the status of the genomic regions. The thresholds are given in the following parameters:

- deltaN
- forceGL
- deletion
- amplicon

Comparing **Figure 2** and **Figure 3** shows the influence of two different sets of parameters.

```
> data(veltman)
> profileCGH <- as.profileCGH(P9)
> profileCGH <- daglad(profileCGH, mediancenter = FALSE, normalrefcenter = FALSE,
+     genomestep = FALSE, smoothfunc = "lawsglad", lkern = "Exponential",
+     model = "Gaussian", qlambda = 0.999, bandwidth = 10, base = FALSE,
+     round = 1.5, lambdabreak = 8, lambdaclusterGen = 40, param = c(d = 6),
+     alpha = 0.001, msize = 5, method = "centroid", nmin = 1,
+     nmax = 8, amplicon = 1, deletion = -5, deltaN = 0.1, forceGL = c(-0.15,
+         0.15), nbsigma = 3, MinBkpWeight = 0.35, CheckBkpPos = TRUE)
```

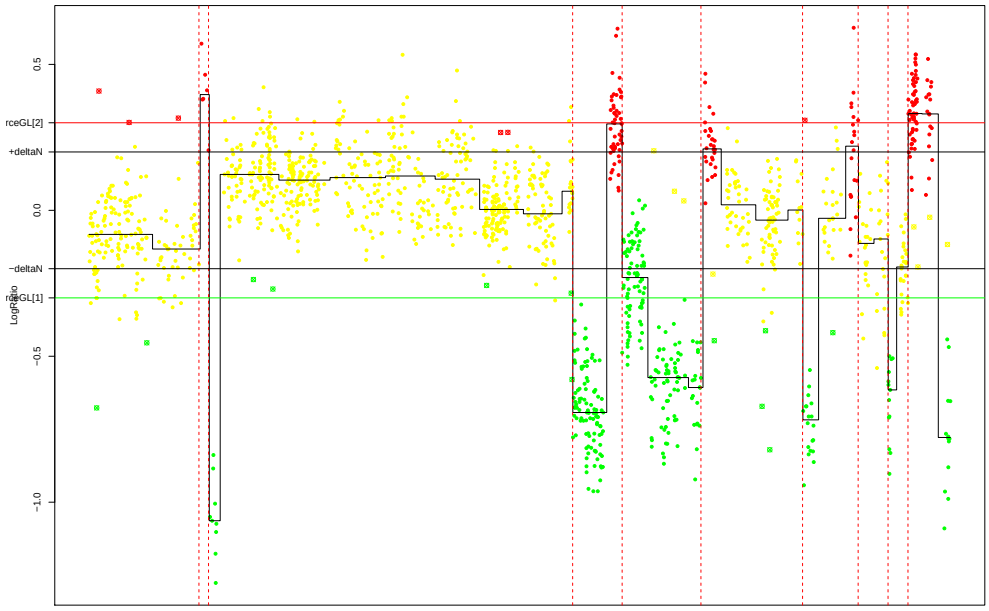

Figure 2: Results of daglad on the patient P9 (Veltman data).

```
[1] "Smoothing for each Chromosome"
[1] "Optimization of the Breakpoints"
[1] "DNA copy number calling"
[1] "subsetdata"
   user  system elapsed
  0.000   0.000   0.001
[1] "diff"
Time difference of 0.003217936 secs
[1] "Temps findCluster: 0.170078039169312"
[1] "jointure BkpInfo"
   user  system elapsed
  0.000   0.000   0.002
[1] "Check Breakpoints Position"
[1] "Results Preparation"
```

The *daglad* function allows a smoothing step over the whole genome (if *genomestep=TRUE*) where all the chromosomes are concatenated together. During this step, the cluster which corresponds to the Normal DNA level is identified: the thresholds used in the function (deltaN, forceGL, amplicon, deletion) are then compared to the median of this cluster.

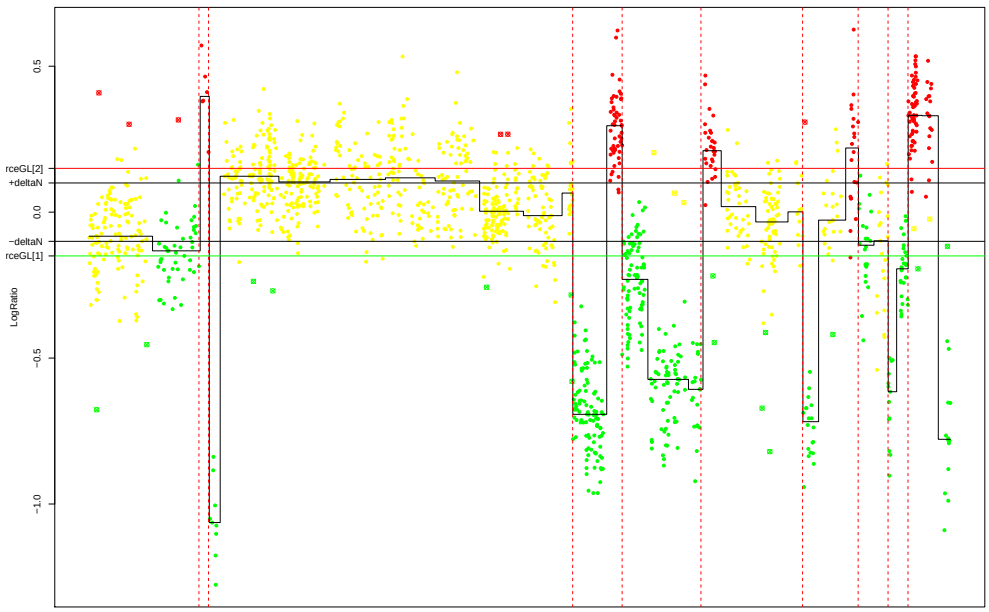

Figure 3: Results of daglad on the patient P9 (Veltman data) - Influence of the thresholds.

4.4 Tuning parameters

The most important parameters are:

- *lambdabreak*
- *lambdacluster*
- *lambdaclusterGen*
- *param* $c(d = 6)$

Decreasing those parameters will lead to a higher number of breakpoints identified. For arrays experiments with very small Signal to Noise ratio it is recommended to use a small value of *param* like $d = 3$ or less.

5 Graphical functions

5.1 Plot of raw array data

```
> data(arrayCGH)
> array <- list(arrayValues = array2, arrayDesign = c(4, 4, 21,
+     22))
> class(array) <- "arrayCGH"
```

```
> arrayPlot(array, "Log2Rat", bar = "none")
```

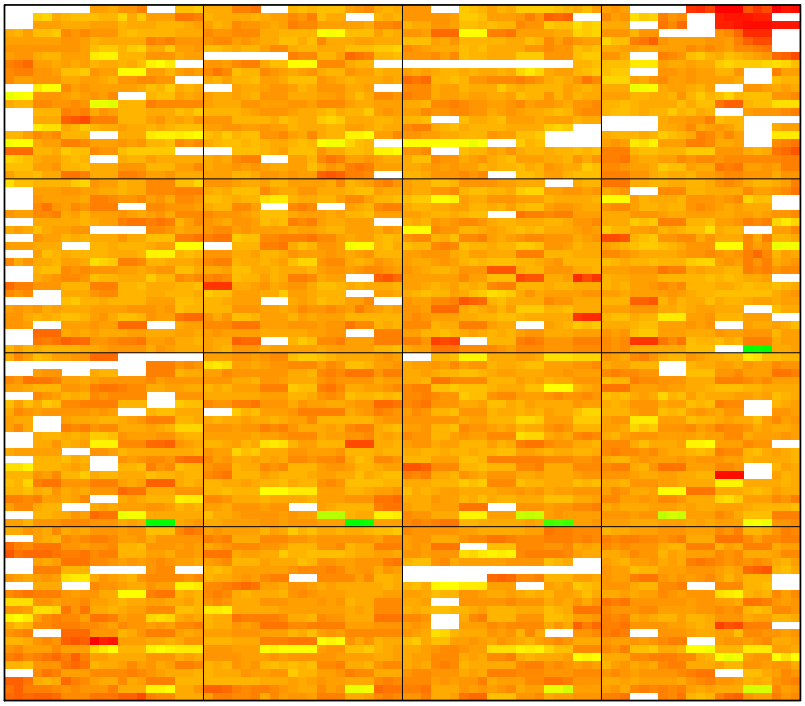

Figure 4: Spatial image of array CGH

```
> arrayPersp(array, "Log2Rat", box = FALSE, theta = 110, phi = 40,
+     bar = FALSE)
```

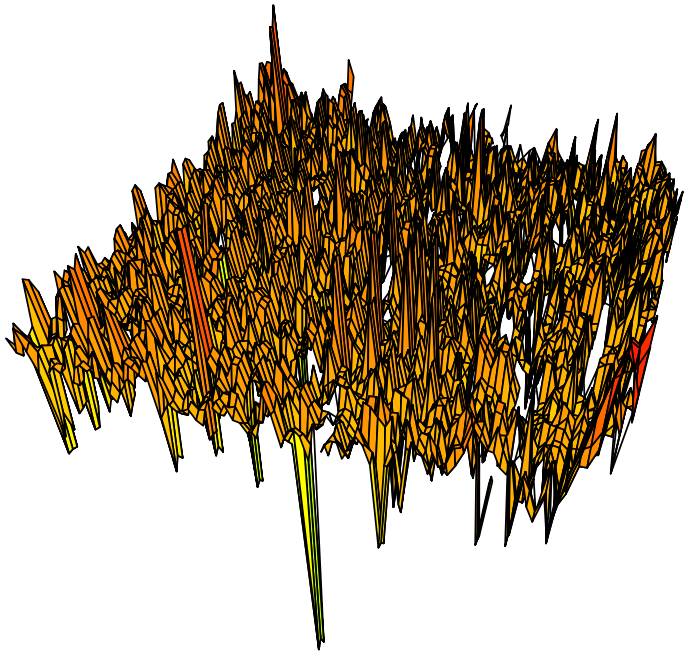

Figure 5: Perspective image of array CGH

5.2 Plot of genomic profile

```
[1] "Smoothing for each Chromosome"
[1] "Optimization of the Breakpoints"
[1] "subsetdata"
   user  system elapsed
      0       0       0
[1] "diff"
Time difference of 0.001724958 secs
[1] "Temps findCluster: 0.0984008312225342"
[1] "subsetdata"
   user  system elapsed
      0       0       0
[1] "diff"
Time difference of 0.001641035 secs
[1] "Temps findCluster: 0.0249760150909424"
[1] "subsetdata"
   user  system elapsed
  0.000   0.000   0.001
[1] "diff"
Time difference of 0.001639128 secs
[1] "Temps findCluster: 0.0249271392822266"
[1] "subsetdata"
   user  system elapsed
      0       0       0
[1] "diff"
Time difference of 0.001827955 secs
[1] "Temps findCluster: 0.102873086929321"
[1] "subsetdata"
   user  system elapsed
      0       0       0
[1] "diff"
Time difference of 0.001636982 secs
[1] "Temps findCluster: 0.0250010490417480"
[1] "subsetdata"
   user  system elapsed
  0.000   0.000   0.001
[1] "diff"
Time difference of 0.001644850 secs
[1] "Temps findCluster: 0.025144100189209"
[1] "subsetdata"
   user  system elapsed
      0       0       0
[1] "diff"
Time difference of 0.001667976 secs
[1] "Temps findCluster: 0.0250189304351807"
[1] "subsetdata"
   user  system elapsed
      0       0       0
[1] "diff"
```

```
Time difference of 0.001652002 secs
[1] "Temps findCluster: 0.0251231193542480"
[1] "subsetdata"
   user  system elapsed
  0.000   0.000   0.001
[1] "diff"
Time difference of 0.001653910 secs
[1] "Temps findCluster: 0.0250368118286133"
[1] "subsetdata"
   user  system elapsed
      0       0       0
[1] "diff"
Time difference of 0.001976967 secs
[1] "Temps findCluster: 0.0253279209136963"
[1] "subsetdata"
   user  system elapsed
      0       0       0
[1] "diff"
Time difference of 0.001996040 secs
[1] "Temps findCluster: 0.127243995666504"
[1] "subsetdata"
   user  system elapsed
      0       0       0
[1] "diff"
Time difference of 0.001667976 secs
[1] "Temps findCluster: 0.0269792079925537"
[1] "subsetdata"
   user  system elapsed
  0.004   0.000   0.001
[1] "diff"
Time difference of 0.001653910 secs
[1] "Temps findCluster: 0.0260670185089111"
[1] "subsetdata"
   user  system elapsed
      0       0       0
[1] "diff"
Time difference of 0.001761913 secs
[1] "Temps findCluster: 0.0271241664886475"
[1] "subsetdata"
   user  system elapsed
  0.000   0.000   0.001
[1] "diff"
Time difference of 0.001682997 secs
[1] "Temps findCluster: 0.0278298854827881"
[1] "subsetdata"
   user  system elapsed
      0       0       0
[1] "diff"
Time difference of 0.001760960 secs
[1] "Temps findCluster: 0.0283198356628418"
```

```
[1] "subsetdata"
   user  system elapsed
      0       0       0
[1] "diff"
Time difference of 0.001681089 secs
[1] "Temps findCluster: 0.0282580852508545"
[1] "subsetdata"
   user  system elapsed
      0       0       0
[1] "diff"
Time difference of 0.001840115 secs
[1] "Temps findCluster: 0.0282530784606934"
[1] "subsetdata"
   user  system elapsed
      0       0       0
[1] "diff"
Time difference of 0.001653910 secs
[1] "Temps findCluster: 0.0253870487213135"
[1] "subsetdata"
   user  system elapsed
      0       0       0
[1] "diff"
Time difference of 0.001650095 secs
[1] "Temps findCluster: 0.0256268978118896"
[1] "subsetdata"
   user  system elapsed
  0.000   0.000   0.001
[1] "diff"
Time difference of 0.001749039 secs
[1] "Temps findCluster: 0.0281779766082764"
[1] "subsetdata"
   user  system elapsed
      0       0       0
[1] "diff"
Time difference of 0.001650095 secs
[1] "Temps findCluster: 0.025651216506958"
[1] "subsetdata"
   user  system elapsed
  0.000   0.000   0.001
[1] "diff"
Time difference of 0.001717091 secs
[1] "Temps findCluster: 0.0263800621032715"
[1] "subsetdata"
   user  system elapsed
  0.000   0.000   0.001
[1] "diff"
Time difference of 0.003642082 secs
[1] "Temps findCluster: 0.183836936950684"
[1] "Results Preparation"
```

```
> plotProfile(res, unit = 3, Bkp = TRUE, labels = FALSE, Smoothing = "Smoothing",
+     plotband = FALSE)
```

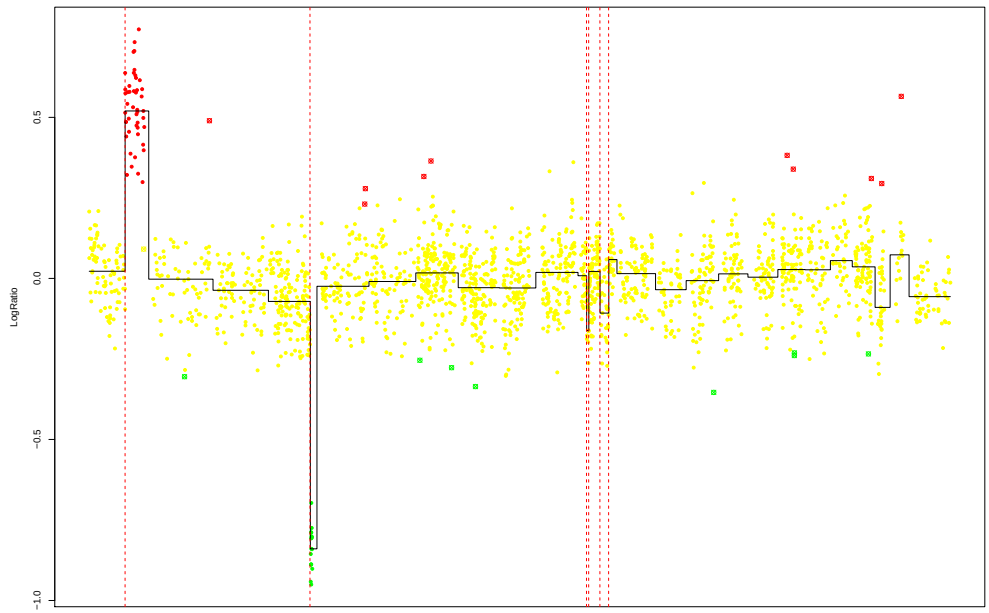

Figure 6: Genomic profile on the whole genome

```
> plotProfile(res, unit = 3, Bkp = TRUE, labels = FALSE, Smoothing = "Smoothing")
```

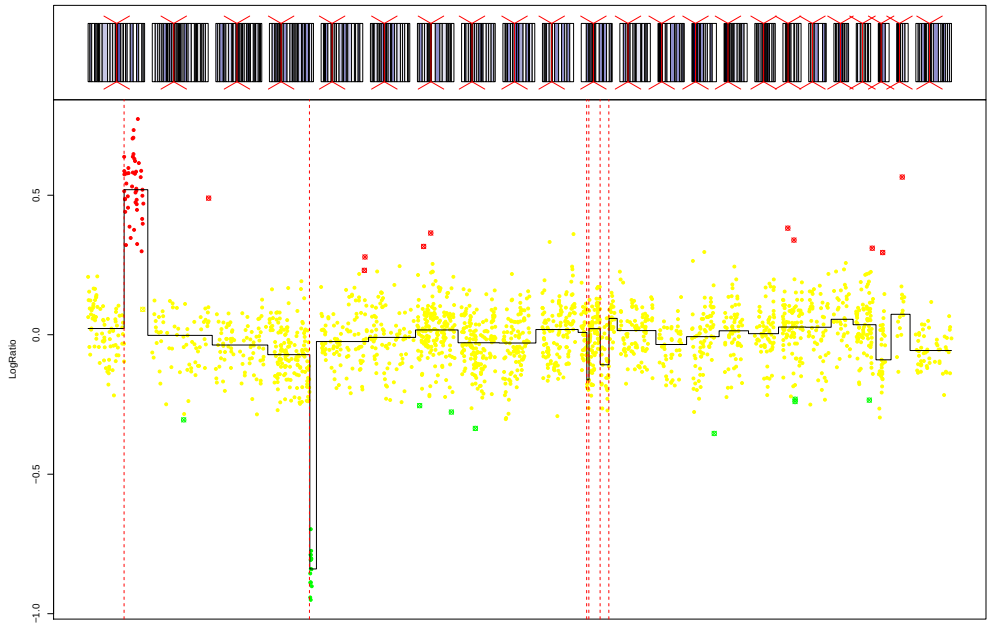

Figure 7: Genomic profile on the whole genome and cytogenetic banding

```
> text <- list(x = c(90000, 2e+05), y = c(0.15, 0.3), labels = c("NORMAL",
+     "GAIN"), cex = 2)
> plotProfile(res, unit = 3, Bkp = TRUE, labels = TRUE, Chromosome = 1,
+     Smoothing = "Smoothing", plotband = FALSE, text = text)
```

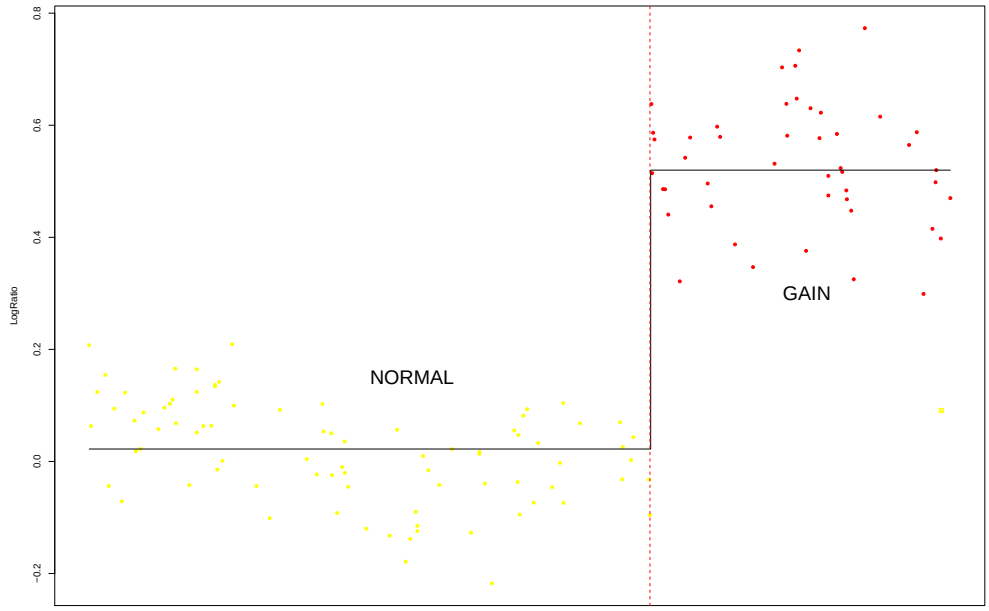

Figure 8: Genomic profile for chromosome 1

```
> text <- list(x = c(90000, 2e+05), y = c(0.15, 0.3), labels = c("NORMAL",
+     "GAIN"), cex = 2)
> plotProfile(res, unit = 3, Bkp = TRUE, labels = TRUE, Chromosome = 1,
+     Smoothing = "Smoothing", text = text, main = "Chromosome 1")
```

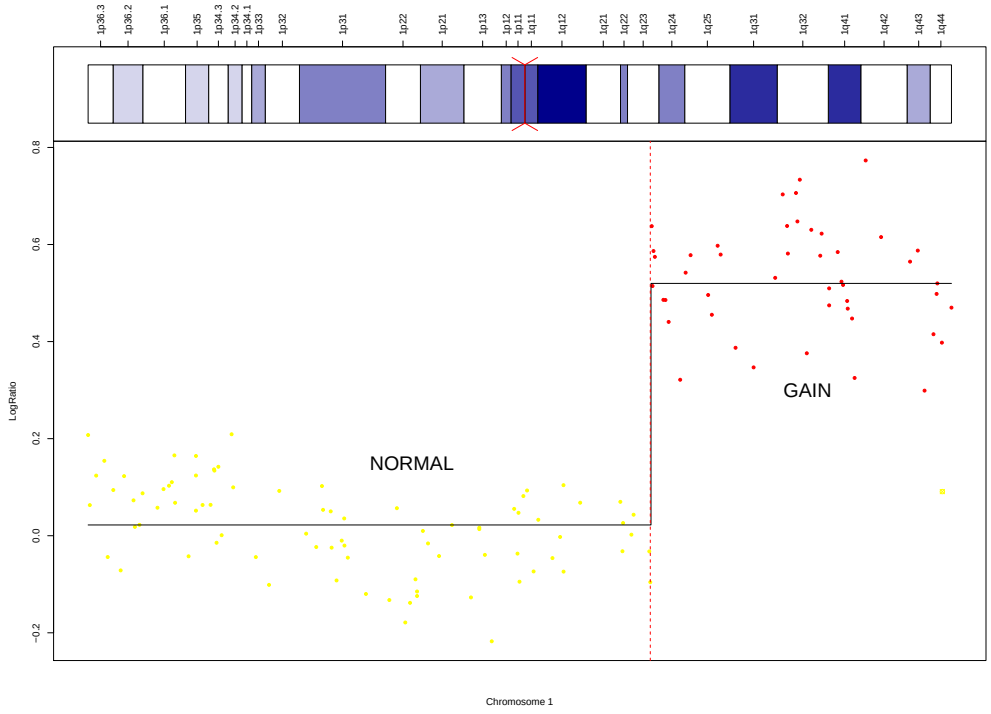

Figure 9: Genomic profile for chromosome 1 and cytogenetic banding with labels

References

Ben-Yaacov, E. and Eldar, Y. C. (2008). A fast and flexible method for the segmentation of acgh data. *Bioinformatics*, 24:i139–i145.

Billerey, C., Chopin, D., Aubriot-Lorton, M. H., Ricol, D., de Medina, S. G. D., Rhijn, B. V., Bralet, M. P., Lefrere-Belda, M. A., Lahaye, J. B., Abbou, C. C., Bonaventure, J., Zafrani, E. S., van der Kwast, T., Thiery, J. P., and Radvanyi, F. (2001). Frequent FGFR3 mutations in papillary non-invasive bladder (pTa) tumors. *Am. J. Pathol.*, 158:955–1959.

Hupé, P., Stransky, N., Thiery, J. P., Radvanyi, F., and Barillot, E. (2004). Analysis of array CGH data: from signal ratio to gain and loss of DNA regions. *Bioinformatics*, 20:3413–3422.

Ishkanian, A. S., Malloff, C. A., Watson, S. K., DeLeeuw, R. J., Chi, B., Coe, B. P., Snijders, A., Albertson, D. G., Pinkel, D., Marra, M. A., Ling, V., MacAulay, C., and Lam, W. L. (2004). A tiling resolution DNA microarray with complete coverage of the human genome. *Nat. Genet.*, 36:299–303.

Pinkel, D., Segraves, R., Sudar, D., Clark, S., Poole, I., Kowbel, D., Collins, C., Kuo, W. L., Chen, C., Zhai, Y., Dairkee, S. H., Ljung, B. M., Gray, J. W., and Albertson, D. G. (1998). High resolution analysis of dna copy number variation using comparative genomic hybridization to microarrays. *Nat. Genet.*, 20:207–211.

Polzehl, J. and Spokoiny, S. (2000). Adaptive weights smoothing with applications to image restoration. *J. Roy. Statistical Society, Series B*, pages 335–354.

Polzehl, J. and Spokoiny, S. (2002). Local likelihood modelling by adaptive weights smoothing. WIAS-Preprint 787.

Snijders, A. M., Nowak, N., Segraves, R., Blackwood, S., Brown, N., Conroy, J., Hamilton, G., Hindle, A. K., Huey, B., Kimura, K., S, S. L., Myambo, K., Palmer, J., Ylstra, B., Yue, J. P., Gray, J. W., Jain, A. N., Pinkel, D., and Albertson, D. G. (2001). Assembly of microarrays for genome-wide measurement of dna copy number. *Nat. Genet.*, 29:263–4.

Solinas-Toldo, S., Lampel, S., Stilgenbauer, S., Nickolenko, J., Benner, A., Dohner, H., Cremer, T., and Lichter, P. (1997). Matrix-based comparative genomic hybridization: Biochips to screen for genomic imbalances. *Genes Chromosomes Cancer*, 20:399–407.

Veltman, J. A., Fridlyand, J., Pejavar, S., Olshen, A. B., Korkola, J. E., DeVries, S., Carroll, P., Kuo, W.-L., Pinkel, D., Albertson, D., Cordon-Cardo, C., Jain, A. N., and Waldman, F. M. (2003). Array-based comparative genomic hybridization for genome-wide screening of DNA copy number in bladder tumors. *Cancer Res*, 63:2872–2880.